Storage Device Performance Prediction with CART Models

Mengzhi Wang, Kinman Au, Anastassia Ailamaki,
Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Storage device performance prediction is a key element of self-managed storage systems and application planning tasks, such as data assignment. This work explores the application of a machine learning tool, CART models, to storage device modeling. Our approach predicts a device's performance as a function of input workloads, requiring no knowledge of the device internals. We propose two uses of CART models: one that predicts per-request response times (and then derives aggregate values) and one that predicts aggregate values directly from workload characteristics. After being trained on the device in question, both provide accurate black-box models across a range of test traces from real environments. Experiments show that these models predict the average and 90th percentile response time with an relative error as low as 19%, when the training workloads are similar to the testing workloads, and interpolate well across different workloads.

1 Introduction

The costs and complexity of system administration make automation of administration tasks a critical research challenge in storage systems [15, 5]. One important aspect of self-managed storage systems, particularly for large storage infrastructures, is deciding which data sets to store on which devices. To find the optimal or a near optimal solution requires the ability to predict how well each device will serve each workload, so that load can be balanced and particularly good matches can be exploited.

Researchers have long utilized performance models for such prediction to compare alternative designs. Given sufficient effort and expertise, an accurate simulation (e.g., [3, 9]) or analytic models(e.g., [7, 10, 11]) can be generated to explore design questions for a particular device. Unfortunately, in practice, such time and expertise is not available for deployed infrastructures. Deployed infrastructures are often comprised of numerous and distinct device types, and their administrators have neither the time nor the expertise needed to configure device models.

This paper attacks this obstacle by providing a black-box model generation algorithm. By "black box," we mean that the model (and model generation system) has no information about the internal components or algorithms of the storage device. Given access to a device for some "training period," the model generation system learns its behavior as a function of input workloads. The resulting device model approximates this function. Our approach takes advantage of an existing machine learning tool, Classification And Regression Trees (CART), to approximate the function because of its efficiency and accuracy. CART models, in a nutshell, approximate functions on a multi-dimensional Cartesian space using piece-wise constant functions.

Such learning-based black box modeling is difficult for two reasons. First, all the machine learning tools we have examined use vectors of scalars as input. Existing workload characterization models, however, involve parameters of empirical distributions. Compressing these distributions into a set of scalars is not straightforward. Second, the quality of the generated models highly depends on the quality of the training workloads. The training workloads should be diverse enough to provide high coverage of the input space.

This work develops two ways of encoding workloads as vectors: a vector per request or a vector per workload. The two encoding schemes lead to two types of device models, operating at the per-request and per-workload granularities, respectively. The request-level device models predict each request's response time based on its per-request vector, or "request description." The workload-level device models, on the other hand, predicte aggregate performance directly from per-workload vectors, or "workload descriptions." Our experiments on a variety of real world workloads have shown that these descriptions are reasonably good in capturing workload performance on both a single disk and a disk array. The two CART-based models have a median relative error of 19% and 47%, respectively, for average response time prediction, and 19% and 50% respectively for the 90th percentile, when the training and testing traces come from the same workload. The CART-based models also interpolate well across workloads.

2 Related Work

Performance modeling has a long and successful history. Almost always, however, thorough knowledge of the system being modeled is assumed. Disk simulators, such as Pantheon [14] and DiskSim [3], simulate storage device behavior by software and produce accurate per-request response times. Developing such simulators is challenging, especially when disk parameters are not publicly available. Predicting performance using simulators is also resource intensive. Analytical models [4, 7, 8, 10, 11]

are more computationally efficient because these models describe device behavior with a set of formulae. Finding the formula set requires deep understanding of the interaction between storage devices and workloads. In addition, both disk simulators and analytical models are tightly coupled with the modeled device. Therefore, new device technologies may invalidate existing models and require a new round of model building.

Our approach treats storage devices as black boxes. As a result, the model construction algorithm is fully automated and should be general enough to handle any type of storage device. The degenerate form of "black-box models" is performance specifications published by device manufacturers, such as the maximum throughput of the devices. The actual performance, however, could be nowhere near these numbers under some workloads. Anderson's "table-based" approach [1] includes workload characteristics in the model input. The table-based models remember the device behavior for a wide range of workload and device pairs and interploates among tables entries in predicting. Our approach improves on the table-based models by employing machine learning tools to capture device behavior. Because of the good scalability of the tools to high dimensional datasets, we are able to use more sophisticated workload characteristics as the model input. As a result, the models are more efficient in both computation and storage.

3 Background: CART Models

This section gives a brief introduction of the CART models and justifies our choice of the tool. A detailed discussion of CART is available in [2].

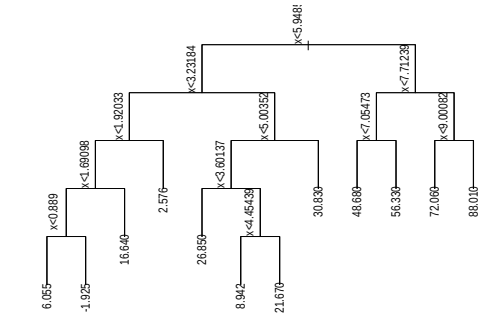
3.1 CART Models

CART models are machine learning tool that can approximate real functions on multi-dimensional Cartesian space. Such tools are also known as regression tools. Given a function $Y = f(X) + \epsilon$, where $X \in \Re^d$, $Y \in \Re$, and ϵ is zero-mean noise, a CART model approximates Y using a piece-wise constant function, $\hat{Y} = \hat{f}(X)$. We refer to the dimensions of X as features. The term, ϵ, captures the intrinsic randomness of the data and the variability contributed by the unobservable variables. The variance of the noise could be dependent on X. For example, the variance of response time often depends on the arrival rates.

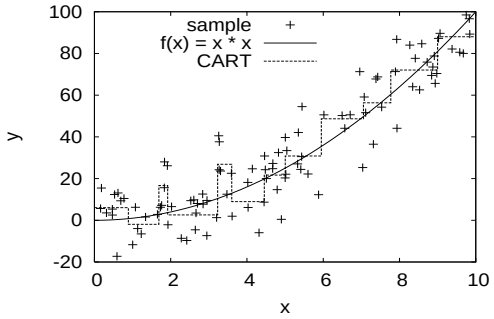
The piece-wise constant function $\hat{f}(X)$ can be visualized as a binary tree. Figure 1(a) shows a CART model constructed on the sample one-dimensional data set in (b). The sample data set is generated using

$$y_i = x_i^2 + \epsilon_i, \qquad i = 1, 2, \ldots, 100.$$

where x_i is uniformly distributed within (0,10), and ϵ_i follows a Guassian distribution of $N(0, 10)$. The leaf nodes correspond to disjoint hyper-rectangles in the feature vector space. The hyper-rectangles are degenerated into intervals for one-dimensional data sets. Each leaf is associated with a value, $\hat{f}(X)$, which is the prediction for all Xs within the corresponding hyper-rectangle. The internal nodes contain split points, and a path from the root to a leaf defines the hyper-rectangle of the leaf node. The tree, therefore, represents a piece-wise constant function on the feature vector space. Figure 1(b) shows the regression line of the sample CART model.



(a) Fitted tree



(b) Data points and regression line

Figure 1: CART model for a simple one-dimensional data set. The data set contains 100 data points generated using $f(x) = x^2 + \epsilon$, where ϵ follows a Guassian distribution with mean 0 and standard deviation 10.

3.2 CART Model Properties

Constructing CART models is efficient. The algorithm starts with an empty tree and grows the tree by greedily selecting the split point that yields the maximum reduction in mean squared error. Each prediction involves one tree traversal and is computationally efficient.

Good interpretability is another desired property provided by CART, because we are interested in the importance of various workload characteristics in predicting workload performance. First, a CART model is a binary tree, making it easy to plot on paper as in Figure 1(a). Second, and more importantly, one can evaluate the importance of a feature by its contribution in error reduction. Intuitively, more important feature should contribute more to the error reduction; thus, leaving it out of the feature vector would significantly raise the prediction error. In a CART model, we can measure the contribution of a feature by summing its contribution in error reduction for all its appearances in the CART model.

4 Predicting Performance with CART

This section presents two ways of constructing device models based on CART models.

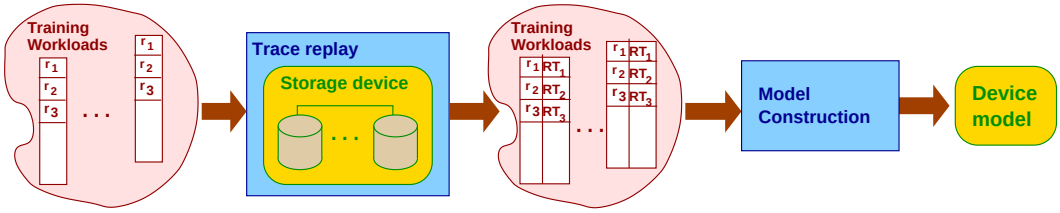


Figure 2: Model construction through training. RT_i is the response time of request r_i.

4.1 Overview

The goal is to build a model for a given storage device to predict device performance as a function of I/O workload. The device model receives a workload as input and predicts its aggregate performance. We define a workload as a sequence of disk requests, with each request, r_i, uniquely described by four attributes: arrival time ($ArrivalTime_i$), logical block number (LBN_i), request size in number of disk blocks ($Size_i$), and read/write type (RW_i). The storage device could be a single disk, a disk array, or some other like-interfaced component. The aggregate performance can be the average and 90-th percentile response time.

Our approach uses CART to approximate the function. We assume that the model construction algorithm can feed any workloads into the device to observe its behavior for a certain period of time, also known as "training." The algorithm then builds the device model based on the observed response times, as illustrated in Figure 2. Model construction does not require any information about the internals of the modeled device, therefore, is general enough to model any devices.

Regression tools are a natural choice to model device behavior. Such tools are designed to model functions on multi-dimensional space given a set of samples with known output. The difficulty is to transform workloads into data points in a multi-dimensional feature space. We explore two ways to achieve the transformation as illustrated in Figure 3. A request-level model represents request r_i as a vector R_i, also known as the "request description," and uses CART models to predict per-request response times. The aggregate performance is then calculated by aggregating the response times. A workload-level model, on the other hand, represents the entire workload as a single vector W, or the "workload description," and predicts the aggregate performance directly from the workload description. In both approaches, the quality of the input vectors is critical to the model accuracy. The next two sections present the request and workload descriptions in detail.

4.2 Request-Level Predictor

This section describes the CART-based request-level device model. This model uses a CART to predict the response times of individual requests based on request descriptions. The model, therefore, is able to generate the entire response time distribution and output any aggregate performance measures.

Our request description R_i for request r_i contains the following features.

$$R_i = \{\ TimeDiff_i(1),\ \ldots,\ TimeDiff_i(k),\ LBN_i,\ LBNDiff_i(1),\ \ldots,\ LBNDiff_i(l),\ Size_i,\ RW_i,\ Seq(i)\ \},$$

where $TimeDiff_i(k) = ArrivalTime_i - ArrivalTime_{i-2^k}$ and $LBNDiff_i(l) = LBN_i - LBN_{i-l}$. The four groups of features capture three components of the response time, and sequentiality of the request. The first $(k+1)$ features measure the temporal burstiness of the workload when r_i arrives, and support prediction of the queuing time. We allow $TimeDiff$ features to look exponentially back to accommodate large bursts. The next $(l+1)$ features measure the spatial locality, supporting prediction of the seek time. The last three support prediction of the data transfer time. The last group has one feature, $Seq(i)$, indicating whether the current request is sequential access.

The two parameters, k and l, determines how far we look back for request bursts and locality. Small values do not adequately capture these characteristics, leading to inferior device models. Large values, on the other hand, leads to a higher dimensionality, meaning a longer training time and larger training set. Our experiments indicate that model accuracy is more sensitive to k than to l, especially for busy workloads, because the queuing time dominates the response time for such workloads.

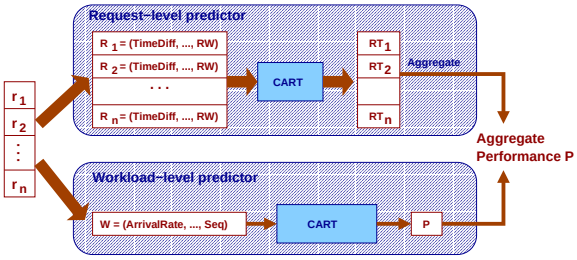


Figure 3: Two types of CART-based device models.

4.3 Workload-Level Device Models

The workload-level model represents the entire workload as a single workload description and predicts aggregate performance based on the description parameters. The workload description W contains the following features.

$$W = \{\ \text{Average arrival rate},$$

Read ratio,
Average request size,
Percentage of sequential requests,
Temporal burstiness,
Spatial burstiness,
Correlations between pairs of attributes }.

The workload description uses the entropy plot [12] to quantify temporal and spatial burstiness and correlations between attributes. The entropy plot plots entropy value on one or two attributes against entropy calculation granularity. The slope of an entropy plot characterizes the degree of burstiness and correlation changes at different granularities. Because of the self-similarity of I/O workloads [6], the entropy plot is usually linear, allowing us to use the slope to characterize the burstiness and correlation. Please refer to [13] for a detailed description of the entropy plot.

Workload-level device models offer fast predictions. The model compresses a workload into a workload description and feeds the description into a CART model to produce the desired performance measure. Both the feature extraction and prediction are fast. To predict both the average and 90th percentile response time, the model must have two separate trees, one for each performance metric.

Workload modeling introduces a parameter called "window size." The window size is the unit of performance prediction and, thus, the workload length for workload description generation. For example, we can divide a long trace into one-minute fragments and use the workload-level models to predict the average response time over one-minute intervals. A short window size has several advantages. First, performance problems are usually transient. A "problem" appears when a large burst of requests arrive and disappears quickly after all the requests in the burst are served. A large window size, on the other hand, fails to indentify such transient bottlenecks. Second, fragmenting a training workload produces more samples for training and reduces the required training time, which is determined by the trace replay time. Windows that are too small, however, contain too few requests for the entropy plot to be effective. We use one minute windows in all of our experiments.

4.4 Comparison of Two Types of Models

There is a clear tradeoff between the request-level and workload-level device models. The former are fast in training and slow in prediction, and the latter are the opposite (slow in training and fast in prediction).

One item for future research is to explore the possibility of combining the two types of models to deliver ones that are efficient in both training and predictions.

5 Experimental Results

This section evaluates the CART-based device models presented in the previous section using a range of workload traces. We conduct experiments to evaluate the model accuracy in modeling both a single disk and a disk array.

Trace name	Length	Requests ($\times 10^6$)	Average Size	% of reads
cello92	4 weeks	7.8	12.9 KB	35.4%
cello99a	4 weeks	43.7	7.1 KB	20.9%
cello99b	4 weeks	13.9	118.0 KB	41.6%
cello99c	4 weeks	24.0	8.5 KB	26.4%
SAP	15 minutes	1.1	15.1 KB	99.9%

Table 1: Trace summary.

Traces. We use three sets of real-world traces in this study. Table 1 lists the summary statistics of the edited traces. The first two, *cello92* and *cello99* capture typical computer system research I/O workloads, collected at HP Labs in 1992 and 1999 respectively. We preprocess *cello92* to concatenate the LBNs of the three most active devices from the trace to fill the modeled device. For *cello99*, we pick the three most active devices, among the 23 devices, and label them *cello99a*, *cello99b*, and *cello99c*. The *cello99* traces fit in a 9GB disk perfectly, so no trace editing is necessary. As these traces are long (two months for *cello92* and one year for *cello99*), we report data for a four-week snapshot (5/1/92 to 5/28/92 and 2/1/99 to 2/28/99).

The *SAP* trace was collected on an Oracle database server running SAP ISUCCS 2.5B in a power utility company. The server has more than 3,000 users and the disk accesses reflect retrieving customer's bills for updating and reviewing. Sequential reads dominate the *SAP* trace.

Devices. We model two types of devices: a single disk and a disk array. The single disk is a 9GB Atlas 10K disk with an average rotational latency of 3 milliseconds. The disk array is a RAID 5 disk array consisting of 8 Atlas 10K disks with stripe unit size of 32KB. We replay all the traces on the two devices except the *SAP* traces, which were collected on devices with a larger capacity than the single disk we are modeling.

Evaluation methodology. The evaluation uses the device models to predict the average and 90th percentile response time for one-minute workload fragments. We report the prediction errors using two metrics: absolute error defined as the difference between the predicted and the actual value, $|\hat{Y} - Y|$, and relative error defined as $\frac{|\hat{Y}-Y|}{Y}$.

In aggregate performance prediction, we use the first two weeks of *cello99a* in training because of the trace's relatively rich access patterns. The training workloads yield 19,583 data points for the workload-level device models. Because of the large number of requests, we use uniform sampling of rate 0.01 to reduce the number of training data points for the request-level models. The final training set contains 218,942 requests.

Predictors in comparison. We evaluate our two CART-based device models, denoted as `CART-request` and `CART-workload` in the remaining text, against three predictors.

- `constant` makes predictions using the average or quantile response time of the training trace.
- `periodic` divides a week into $24 \times 7 \times 60$ one-minute intervals and remembers the aggregate performance of the training workload for each interval. Prediction uses the corresponding value of the interval with the same offset within the week.
- `linear` does linear regression on the workload descriptions.

Note that the `constant` and `periodic` predictors model workloads rather than devices, because they do not take workload characteristics as input. Both predictors rely on the similarity between the training and testing workloads to produce accurate predictions. The difference between `linear` and `CART-workload`, on the other hand, shows the importance of using non-linear models, such as the CART models, in device modeling.

5.1 Calibrating Request-Level Models

We evaluate the prediction accuracy of request-level models in this section.

Figure 4 shows the relative importance of the parameters of the request description in determining per-request response time. The relative importance of a parameter is measured by its contribution in error reduction. We show the importance measured on two traces, *cello99a* and *cello99c*. Because of the large number of the requests, we use only the first day of the traces and reduce the data set size by 90% with uniform sampling.

First, we observe that the relative importance is workload dependent. As we expected, for busy traffic such as *cello99a*, the queuing time dominates the response time, and thereby, the $TimeDiff$ parameters are more important. *cello99c*, on the other hand, has small response times, and parameters that characterize the data transfer time, such as $Size$ and RW, have good predictive power.

Second, the most imporant parameter shifts from $TimeDiff8$ to $TimeDiff7$ from the single disk to the disk array for *cello99a* because the queuing time becomes less significant for the disk array. The distinction between the two traces, however, persists.

We select to use a history length of 10 for $TimeDiff$ and 3 for $LBNDiff$ in the subsequent experiments so that we can model device behavior under both types of workloads.

Figure 5 compares the predicted response time distribution against the actual one. The long tail of the actual response time distribution is well captured by the request-level model. The model has a median absolute error of 4.28 milliseconds, close to the average rotational latency (3 milliseconds) of the modeled disk. The high prediction accuracy indicates that the request description is effective in characterizing request characteristics needed to predict response times.

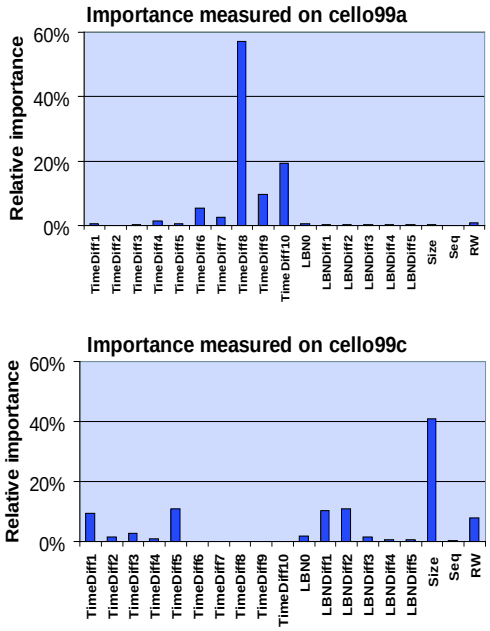


Figure 4: Relative importance of parameters in the request description for the Atlas 10K disk.

In summary, the request description effectively captures important per-request characteristics, leading to accurate request-level device models.

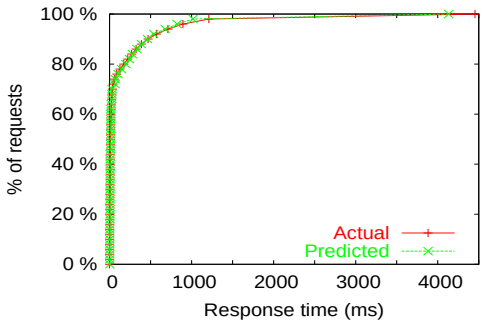
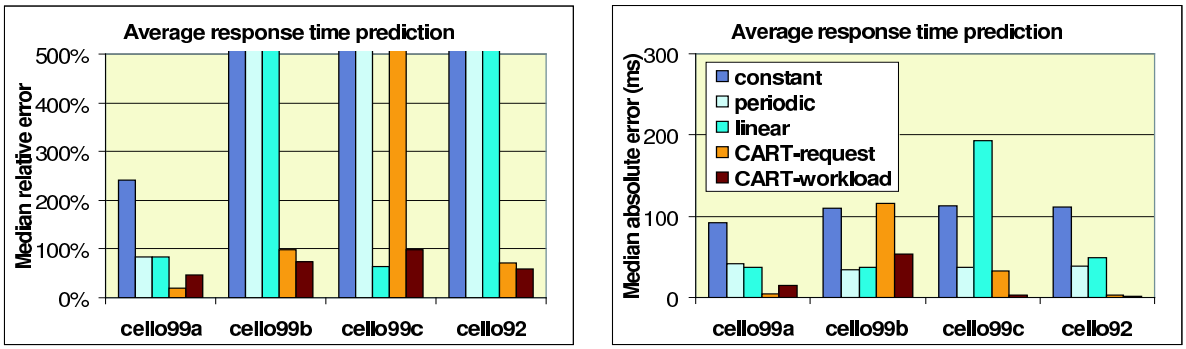


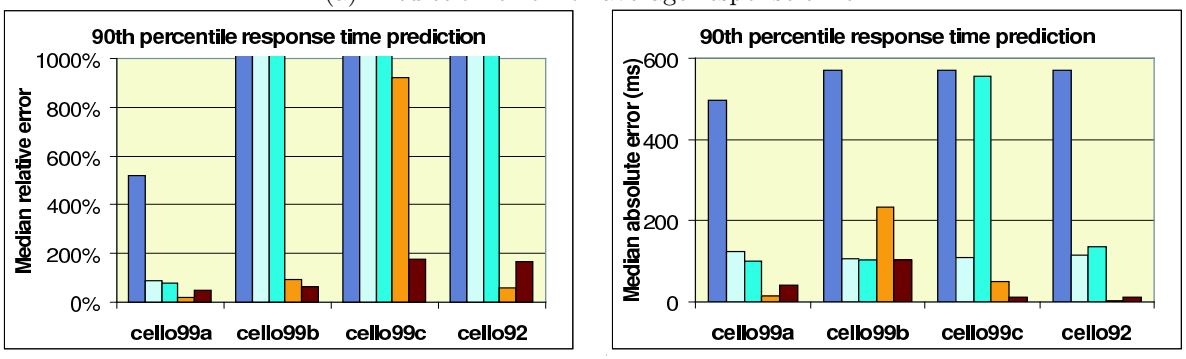
Figure 5: Prediction accurary of the request-level model. The actual and predicted average response times are 137.96 ms and 133.01 ms respectively. The corresponding demerit, defined in [9] as the root mean square of horizontal distance between the actual and predicted curves in (b), is 46.06 milliseconds (33.4%).

5.2 Modeling A Single Disk

Figure 6 compares the accuracy of all the predictors in modeling an Atlas 10K 9GB disk. As mentioned earlier, all the predictors are trained using the first two weeks of *cello99a*. Overall, the two CART-based device models provide good prediction accuracies in predicting both the average and 90th percentile response times, compared to other predictors. Several more detailed observations can



(a) Prediction error for average response time



(b) Prediction error for 90th percentile response time

Figure 6: Comparison of predictors for a single 9GB Atlas 10K disk.

be made.

First, all of the models perform the best when the training and testing workloads are from the same trace, *cello99a*, because the models have seen how the device behaves under such workloads. The `periodic` predictor also cuts the median prediction error of the `constant` predictor by more than a half because the strong periodicity of the workload. `CART-request` and `CART-workload` further reduce the error to 4.81 milliseconds (19%) and 14.83 milliseconds (47%) respectively for the average response time prediction, and 15.38 milliseconds (19%) and 42.63 milliseconds (50%) respectively for the 90th percentile. The performance difference between `linear` and `CART-workload` roughly measures the benefit of using a non-linear model, such as CART, because both take the same input. We observe a significant improvement from the former to the latter, suggesting non-linear device behavior.

Second, both CART-based device models provide better interpolation across workloads than the other models. `constant` and `periodic` rely blindly on similarity between the training and testing workloads to make good predictions. Consequently, it is not surprising to see huge prediction errors when the training and testing workloads differ. The CART-based predictors, on the other hand, distinguish workloads of different characteristics, thereby, providing better interpolation. The large relative error for `cello99c` is due to the small average response time of the trace.

Third, model accuracy is highly dependent on the training workload quality for the CART-based models. The prediction error increases for workloads other than *cello99a*, because of the access pattern differences among these traces. The CART-based models learn device behavior through training; therefore, they can not predict workload performance for ones that have totally different characteristics from the training workloads. For example, `CART-request` constantly over-predicts for *cello99c* because the model was never trained wth the small sequential accesses that are popular in `cello99c`. Section 5.4 gives an informal error analysis and identifies inadequate training being the most significant error source.

Fourth, high quantile response times are more difficult to predict. We observe larger prediction errors from all the predictors for 90th percentile response time predictions than for average response tmie predictions. The accuracy advantage of the two CART-based models is higher for 90th percentile predictions.

In summary, the two CART-based models give accurate predictions when the training and testing workloads share the same characteristics and provide better interpolation otherwise. The good accuracy suggests the effectiveness of the request and workload descriptions in capturing important workload characteristics. The training workload quality plays an important role in the model accuracy for CART-based models.

5.3 Modeling A Disk Array

Figure 7 compares the accuracy of the five predictors in modeling a disk array except `periodic` because the

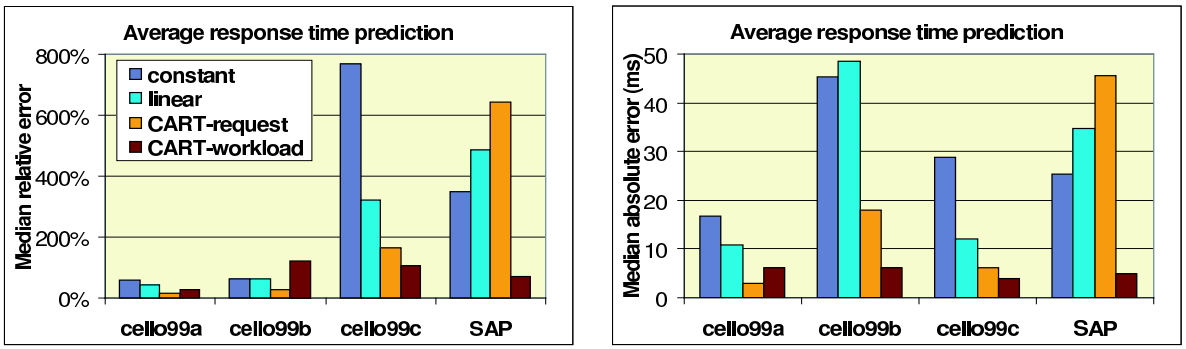


Figure 7: Comparison of predictors for a RAID 5 disk array of 8 Atlas 10K disks in predicting average response time.

SAP traces do not provide enough information on arrival time for us to know the offset within a week. The overall results are similar to those for the single disk. The two CART-based models are the most accurate predictors when the training and testing workloads are from the same trace. Also, due to the decreased response time from the single disk to the disk array, the absolute errors become smaller, and the relative errors become larger. The relative accuracy among the predictors, however, stays the same. In particular, the CART-based models perform better on all the workloads in general. Overall, the CART-based device modeling approach works well for the disk array.

5.4 Error Analysis

We conduct an experiment to identify error sources of the CART-based device models. A model's error consists of two parts. The first part comes from intrinsic randomness of the input data, such as measurement error, and this error can not be captured by any model. The rest of the error comes from the modeling approach itself. The CART-based models incur error at three places. First, the transformation from workloads to vectors introduces information loss. Second, the CART-based models use piece-wise constant functions, which could be different from the true functions. Third, a low-quality training trace yields inaccurate models because CART relies on the information from the training data to make predictions. We usually find that the last one, inadequate training data, causes the most trouble. An inadequate training set has only a limited range of workloads and leads to large prediction errors for workloads outside of this range.

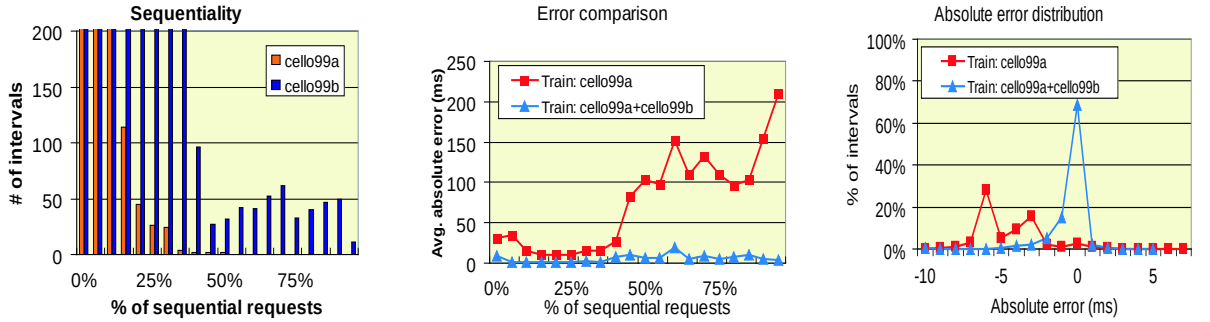
We conduct a small experiment to verify our hypothesis. Figure 8 (a) compares the difference in sequentiality between *cello99a* and *cello99b*. The spectrum of sequentiality (from 0% to 100% of requests in the workload being sequential) is divided into 20 buckets, and the graphs shows the number of one-minute workload fragments in each bucket for both traces. We observe a significant number of high sequentiality fragments in *cello99b*, but no fragment goes beyond 0.5 sequentiality in *cello99a*. This difference leads to large prediction errors for high sequentiality fragments when we build the model on *cello99a* and predict the performance of *cello99b*, as shown in (b). The errors are reduced significantly when we include the first half of *cello99b* in training. The dramatic error reduction suggests that prediction errors from the other sources are negligible when compared with the ones introduced by inadequate training. Figure 8 (c) further shows the absolute error histogram with 1 millisecond buckets. The spike shift to 0 milliseconds when using the combined training trace, indicates that it is reasonable to assume a zero-mean noise term. We conclude from these results that contributing efforts in black-box device modeling should be directed to generate a good training set that covers a broad range of workload types.

6 Conclusions

Storage device performance modeling is an important element in self-managed storage systems and other application planning tasks. Our target model takes a workload as input and predicts its aggregate performance on the modeled device efficiently and accurately. This paper presents our initial results in exploring machine learning tools to build device models. A black box predictive tool, CART, makes device models independent of the storage devices being modeled, and thus, general enough to handle any type of devices. This paper presents two ways of applying CART models, yielding request-level and workload-level device models. Our experiments on real-world traces have shown that both types of models are accurate and efficient. The error analysis suggests that the quality of the training workloads plays a critical role in model accuracy. Continuing research can improve model prediction accuracies and the efficiency of workload description.

7 Acknowledgments

We thank the members and companies of the PDL Consortium (including EMC, Hewlett-Packard, Hitachi, Hitachi Global Storage Technologies, IBM, Intel, LSI Logic, Microsoft, Network Appliance, Oracle, Panasas, Seagate, Sun, and Veritas) for their interest, insights, feedback, and support. We thank IBM for partly funding this work through a CAS student fellowship and a faculty partnership award. This work is funded in part by NSF grants CCR-0205544, IIS-0133686, BES-0329549,



(a) Histograms on sequentiality (b) Average absolute error (c) Histograms on absolute error

Figure 8: Effects of different training workloads.

IIS-0083148, IIS-0113089, IIS-0209107, and IIS-0205224. We would also like to thank Eno Thereska, Mike Mesnier, and John Strunk for their participation and discussion in the early stage of this project.

References

[1] Eric Anderson. Simple table-based modeling of storage devices. Technical Report HPL-SSP-2001-4, HP Labs, 2001.

[2] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[3] John Bucy, Greg Ganger, and contributors. The DiskSim simulation environment version 3.0 reference manual. Technical Report CMU-CS-03-102, Carnegie Mellon University, 2003.

[4] Shenze Chen and Don Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, 1996.

[5] Gregory R. Ganger, John D. Strunk, and Andrew J. Klosterman. Self-* storage: Brick-based storage with automated administration. Technical Report CMU-CS-03-178, Carnegie Mellon University, 2003.

[6] María E. Gómez and Vicente Santonja. Analysis of self-similarity in I/O workload using structureal modeling. In *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 234–243, 1999.

[7] Edward K. Lee and Randy H. Katz. An analytic performance model of disk arrays. In *Proceedings of the 1993 ACM SIGMETRICS*, pages 98–109, 1993.

[8] Arif Merchant and Guillermo A. Alvarez. Disk array models in Minerva. Technical Report HPL-2001-118, HP Laboratories, 2001.

[9] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, 1994.

[10] Elizabeth Shriver, Arif Merchant, and John Wilkes. An analytical behavior model for disk drives with readahead caches and request reordering. In *Proceedings of International Conference on Measurement and Modeling of Computer Systems*, pages 182–191, 1998.

[11] Mustafa Uysal, Guillermo A. Alvarez, and Arif Merchant. A modular, analytical throughput model for modern disk arrays. In *Proceedings of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 183–192, 2001.

[12] Mengzhi Wang, Anastassia Ailamaki, and Christos Faloutsos. Capturing the spatio-temporal behavior of real traffic data. *Performance Evaluation*, 49(1/4):147–163, 2002.

[13] Mengzhi Wang, Kinman Au, Anastassia Ailamaki, Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger. Storage device performance prediction with CART models. Technical Report CMU-PDL-04-103, Carnegie Mellon University, 2004.

[14] J. Wilkes. The Pantheon storage-system simulator. Technical Report HPL–SSP–95–14, Hewlett-Packard Laboratories, 1995.

[15] John Wilkes. Data services – from data to containers. Keynote address at File and Storage Technologies Conference (FAST'03), March – April 2003.